职教本科计算机系列教材

Vue.js 前端框架项目实战

（微课版）

李　丽　主　编

叶继阳　顾云山　刘　段　副主编

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书是以实战为导向的前端开发教程，以技术社区网站项目为依托，系统讲解 Vue.js 框架的核心技术及现代前端工程化实践，涵盖从基础语法到项目部署的全流程技能。全书共六章：第一章奠定前端框架开发基础，涵盖 Web 前端技术发展概述、前端开发工具、Vue.js 入门、前端 UI 框架入门及版本控制与代码托管；第二章深入基本指令，涵盖数据绑定及相关指令、条件渲染和列表渲染；第三章聚焦交互特性，解析事件处理、计算属性与侦听器；第四章以项目工程化为核心，讲解项目初始化、组件化开发与生命周期；第五章进阶到应用架构优化，探讨路由、组件与状态管理；第六章进行联调与部署实战，完成前后端联调、项目打包部署等生产级操作。

本书可以作为高等职业本专科院校计算机相关专业的教材，也可以作为想要从事 Web 前端开发相关工作的学习者的自学教材，还可以作为 IT 类培训机构的 Web 前端开发培训教材。

图书在版编目（CIP）数据

Vue.js 前端框架项目实战 ： 微课版 / 李丽主编.
北京 ： 电子工业出版社, 2026. 1. -- ISBN 978-7-121-51532-3

Ⅰ. TP393.092.2

中国国家版本馆 CIP 数据核字第 2025XX9903 号

责任编辑：杨永毅
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×1092　1/16　印张：11.75　字数：301 千字
版　　次：2026 年 1 月第 1 版
印　　次：2026 年 1 月第 1 次印刷
印　　数：1 200 册　　定价：45.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）88254570，xujj@phei.com.cn。

前　言

在互联网快速迭代的当下，前端技术已经成为提升用户体验的核心驱动力。Vue.js 凭借其渐进式架构、轻量高效的特点，成为全球开发者最为青睐的前端框架之一。无论是初创项目，还是大型应用，Vue.js 都能以灵活的组件化设计、直观的响应式机制，帮助开发者高效实现复杂功能。本书以 Vue 3.x 版本为核心，结合 Vite、Pinia 等最新技术生态，为读者提供符合行业趋势的学习路径。

本书面向希望系统掌握 Vue.js 并具备全栈视野的开发者。无论是刚接触前端开发的学生，还是希望从传统开发转向现代框架的职业工程师，都能通过本书的阶梯式内容获得提升。本书摒弃空洞的理论堆砌，以“技术社区网站”为实战项目贯穿始终，将知识点融入需求分析、开发调试、性能优化等真实环节，确保读者既能理解原理，又能动手解决实际问题。特别地，本书结合职业本科与高职院校的教学特点，强调“项目中心课程”的教学理念，通过真实项目的开发过程，不仅能使读者掌握 Vue.js 的核心技术，还能培养读者解决实际问题的能力，提升职业素养和团队协作能力。

全书以“基础入门→进阶实战”为主线，六章内容环环相扣。基础入门篇（第一章到第三章）：以“HTML 版技术社区网站”项目为载体，从 Web 前端技术发展切入，详细讲解 Vue.js，包括数据绑定及相关指令、条件渲染、列表渲染等基本指令，以及事件处理、计算属性等交互特性，还引入了 VS Code 高效开发技巧与 Element Plus 前端 UI 框架，通过开发网站首页、注册和登录页面、个人中心页面、商城页面等功能模块夯实基础。进阶实战篇（第四章到第六章）：以“Vite 版技术社区网站”项目为载体，深入工程化实践，讲解如何使用脚手架创建项目、进行组件化开发及合理运用生命周期，接着运用路由、组件、状态管理等进行应用架构优化，并打通开发全链路，从前后端联调到项目打包部署，完整呈现项目上线流程，培养读者解决实际问题的能力。

本书的亮点在于实战驱动的教学模式，每章都配备了“习题”与“任务”，让读者能够在学习过程中即时应用所学知识，实现即学即用的效果。同时，书中融入了现代工具链，包括 AI 代码工具、Git 版本控制工具、Vue DevTools 调试及 ESLint 规范，使读者能够接触贴近企业级开发环境的工具和流程。此外，本书还注重培养读者的全栈思维，不仅关注前端开发，还涉及模拟后端接口、项目打包部署等内容，帮助读者拓宽技术视野，从而更好地适应实际开发中的全流程工作。

本书建议读者在学习过程中秉持“动手为先”原则。书中所有案例均配备完整代码，鼓励读者边学边练，积极尝试优化界面、扩展功能，以及重构代码，从而加深对知识的理解和掌

握。同时，读者要善于利用各种工具来提升学习效率和质量。此外，书中每章都设有“素质目标”，旨在引导读者关注技术发展趋势和协作规范，鼓励读者积极参与开源社区，通过拓展思考提升自身的综合素质和竞争力。前端技术虽日新月异，但其底层逻辑与工程思想历久弥新。希望读者不仅能通过本书掌握 Vue.js 前端框架技术，还能培养独立解决问题的能力，保持持续学习的热情。

本书由金华职业技术大学组织编写，由李丽担任主编，由叶继阳、顾云山、刘段担任副主编。本书的顺利完成得益于中国联合网络通信有限公司金华市分公司和金华市信创评测技术服务有限公司企业工程师的项目实践经验，以及金华职业技术大学邱晓华教授和陈晓龙教授的大力支持和指导，在此深表敬意。

为方便教师教学，本书配有微课视频、教学大纲、教学课件、习题答案、源代码和素材等教学资源，请有此需求的教师登录华信教育资源网，注册后免费下载。如有问题，可在网站留言板留言或与电子工业出版社联系（E-mail：hxedu@phei.com.cn）。与本书配套的数字课程“前端框架技术与应用”已在超星“学银在线”网站上线，读者可登录该网站进行在线学习和资料下载，教师可以调用本课程来构建符合自身教学特色的在线课程。

由于编者水平有限，书中难免存在纰漏，恳请各位同行与读者批评指正。

编　者

目 录

第一章

前端框架开发基础

学习三维目标

知识目标

- 了解 Web 前端技术的发展历程
- 理解 MVVM 前端架构模式
- 理解单页面应用的概念

能力目标

- 能够安装并使用前端开发工具
- 能够渐进式引入 Vue.js 并创建应用实例
- 能够安装前端 UI 框架并合理运用

素质目标

- 强化科技报国使命担当
- 培育自主创新思维
- 培养团队协作与共享意识

1.1 Web 前端技术发展概述

从最初仅能呈现简单文字的静态页面，到如今功能丰富、交互复杂的大前端时代，前端技术经历了三十多年的飞速发展。在这一过程中，技术的迭代与创新从未停歇，推动着互联网体验的不断升级。那么，前端技术是如何一步步演进的？当前的前端生态又呈现出怎样的面貌？接下来，我们将一同探寻前端技术的发展脉络，回顾那些改变互联网历史的关键节点。

1.1.1 前端技术演变

1. Web 1.0 时代：从静态页面到动态页面的跨越

20 世纪 90 年代初，互联网的雏形开始显现。1990 年，蒂姆·伯纳斯–李（Tim Berners-Lee）开发了首个 Web 服务器和客户端，为万维网的诞生奠定了基础。1991 年，他创建了世界上第一个网站，标志着网页正式登上了互联网的舞台。

1993 年，公共网关接口（CGI）出现，首次实现了后端动态生成页面。然而，由于页面内容完全依赖后端生成，每次更新都需要加载整个页面，在当时的网络条件下，这不仅效率低下，还严重影响用户体验。为了改善这一问题，前端技术开始从编写语言、浏览器功能和 HTML 标准等方面进行优化。

1994 年，网景公司推出了首款商业浏览器 Navigator，随后微软发布了 IE 浏览器，拉开了浏览器竞争的序幕。同时，PHP 语言诞生，它能将动态内容直接嵌入 HTML，极大提升了页面开发的效率和可读性，这一创新为后来的 ASP、JSP 及前端模板引擎的发展奠定了基础。与此同时，W3C（万维网联盟）成立，致力于 HTML 标准的制定与推广，为前端技术的规范化铺平了道路。

1995 年，JavaScript 的问世彻底改变了网页的交互方式。它允许开发者在浏览器端动态操作 DOM 元素，使网页无须刷新页面即可更新内容。尽管微软的 IE 浏览器一度占据市场主导地位，但 JavaScript 还是通过 ECMA 组织的标准化，成为全球通用的脚本语言，推动了前端技术的进一步发展。

2. Web 2.0 时代：异步交互与用户体验的革新

2004 年左右，Web 2.0 的概念开始流行。2005 年左右，AJAX 技术开始广泛应用于 Web 开发，谷歌推出了基于 AJAX 技术的 Gmail 和 Google Maps，极大提升了网页浏览体验。

与此同时，HTML 标准也在不断演进。2008 年，W3C 发布了 HTML5 草案，引入了对视频、音频、图像和动画等多媒体内容的原生支持，使网页成为一个更成熟的应用平台。2014 年，HTML5 正式发布，各大浏览器纷纷升级以支持新标准，推动了浏览器技术的良性竞争。

值得一提的是，2008 年，谷歌推出了 V8 JavaScript 引擎，显著提升了浏览器的性能。2009 年，第五代 JavaScript 标准（ECMAScript 5）发布，进一步推动了 JavaScript 的标准化和功能拓展。2010 年，AngularJS（Angular 的第一个版本）诞生，随后被谷歌收购，成为前端开发的重要工具。2013 年，Facebook 推出了 React 框架，凭借其组件化设计和高效的渲染性能，迅速赢得了开发者的青睐。2014 年，Vue.js 问世，其轻量级和渐进式的设计理念，使开发者

能够快速上手，成为适用于各种规模应用的热门框架。

3. 大前端时代：全栈化与生态系统的繁荣

随着前端技术的发展，前端开发逐渐从单纯的页面构建扩展到全栈领域。2009 年，Ryan Dahl 发布了 Node.js，这是一个基于 V8 JavaScript 引擎的服务器端 JavaScript 运行环境。Node.js 的出现，标志着 JavaScript 突破了浏览器的限制，能够用于服务器端开发，还标志着前端技术进入了全栈化的新时代。如果说 AJAX 是前端技术的第一次飞跃，那么 Node.js 就是前端技术的第二次飞跃，它开启了大前端时代。

如今，前端技术已经形成了一个庞大的技术生态系统，涵盖从开发工具到框架的多个领域，包括以下内容。

- 版本控制工具：如 Git，为团队协作和代码管理提供了强大的支持。
- 包管理工具：如 NPM 和 Yarn，简化了依赖管理和模块化开发。
- 脚本语言体系：包括 ECMAScript 6（ES6）、TypeScript 和 Babel，为开发者提供了现代化的语言特性和增强功能。
- 网页基础技术：如 HTML5 和 CSS3，它们的普及使网页能够支持更丰富的交互和视觉效果。
- 前端框架：如 Angular、React 和 Vue.js，能够帮助开发者高效构建复杂的单页面应用。
- 打包工具：如 Webpack 和 Vite，优化了资源管理和项目构建流程。
- 后端框架：基于 Node.js 的 Express 和 Koa，进一步拓展了前端开发者的全栈能力。

这些技术的发展，不仅提升了前端开发的效率，还推动了互联网应用的多样化和复杂化。本书将以 Vue.js 框架为主线，通过实战项目演示，全面讲解前端项目的完整开发流程。

1.1.2 前端架构模式

架构模式是解决软件体系结构常见问题的通用且可复用的方案，其应用范围比软件设计模式更加广泛。遵循规范体系对软件系统设计至关重要，这是因为它能确保系统迭代敏捷、沟通顺畅。因此，在软件开发初期，选择合适的架构模式极为关键，它能保障系统的功能和质量。

随着浏览器对 HTML5 的支持，前端交互变得更加丰富，代码复杂度也随之增加。原本用于后端开发的 MV*架构模式逐渐被引入前端开发。

1. MVC 架构模式

MVC（Model-View-Controller）架构模式是一种经典的软件设计模式，最早于 20 世纪 80 年代提出。它通过将应用程序划分为三个核心模块——模型（Model）、视图（View）和控制器（Controller），实现了业务逻辑、数据和数据展示的分离。

MVC 架构模式由三部分组成，如图 1-1 所示。

- Model：负责存储和管理应用程序的核心数据，如数据库操作。
- View：负责接收用户输入和数据展示，如 HTML 页面。
- Controller：负责处理用户输入和业务逻辑，并协调 Model 和 View 之间的交互。

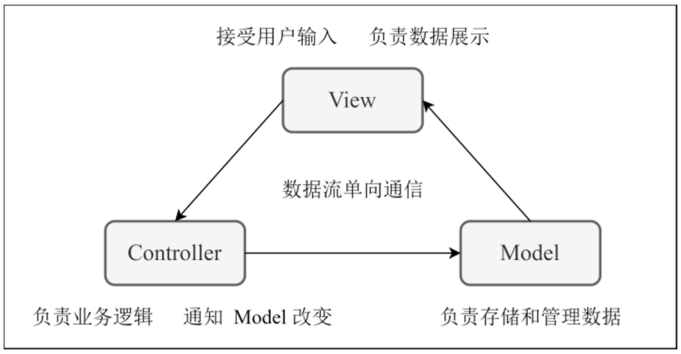

图 1-1　MVC 架构模式

MVC 为开发者提供了对 HTML、CSS 和 JavaScript 的全面控制，通过分离 Model 和 View 的代码，实现了更清晰的职责划分。然而，MVC 是一种单向通信模式，Model 和 View 之间的交互需要借助 Controller 作为中介，Controller 在其中起到了桥梁的作用，负责数据流的传递和协调。

2. MVVM 架构模式

在 MVVM 架构模式出现之前，前端开发面临着诸多挑战，如大量的 DOM 操作导致代码冗余和难以维护、频繁的页面刷新影响性能、手动同步数据状态复杂且容易出错。而 MVVM 架构模式的出现有效解决了这些问题。MVVM（Model-View-ViewModel）架构模式是一种简化用户界面事件驱动编程的架构模式，源自 MVC 和 MVP 架构模式。其核心特点是双向数据绑定，能够自动同步 View 和 ViewModel 的状态，从而减少了手动更新页面的烦琐操作。

MVVM 架构模式由三部分组成，如图 1-2 所示。

- Model（模型）：负责数据和业务逻辑，定义数据的修改和操作。
- View（视图）：UI 组件，负责将数据可视化呈现给用户。
- ViewModel（视图模型）：作为 View 和 Model 之间的桥梁，通过数据双向绑定自动同步数据变化。

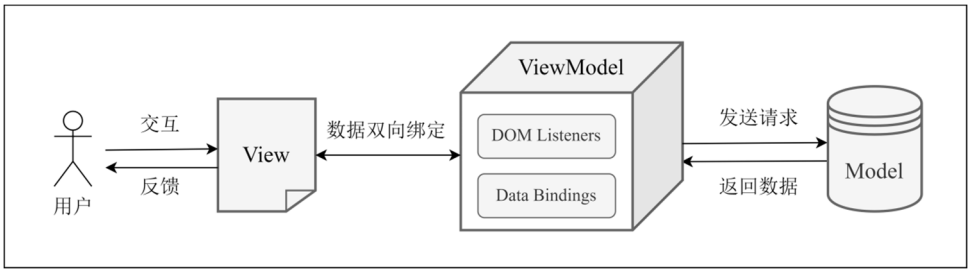

图 1-2　MVVM 架构模式

在 MVVM 架构模式中，View 和 Model 并非直接交互，而是通过 ViewModel 进行数据传递。View 和 Model 之间的同步是自动的，无须开发者手动干预。开发者只需专注于业务逻辑，而 DOM 操作和数据状态的同步则交由框架来管理，从而极大地提高了开发效率。

MVVM 架构模式的引入推动了前端开发与后端业务逻辑的分离，显著提升了前端开发效率。ViewModel 作为中转站，负责将 Model 中的数据转换为易于管理和使用的格式。它通过

双向数据绑定与 View 交互，同时通过接口请求与 Model 通信，实现数据的双向同步。

这种轻量级的架构模式使前端开发更加高效和便捷。当前主流的 JavaScript 框架，如 Angular、React 和 Vue.js，都是 MVVM 架构模式的典型实现。这些框架的应用推动了网站从传统的 Web Site 向更复杂的 Web App 演变，开启了单页面应用时代。

1.1.3 单页面应用

单页面应用（Single-page Application，SPA）是一种创新的 Web 架构模式，它通过动态更新页面内容，为用户带来流畅、无缝的交互体验，避免了传统多页面应用（Multi-page Application，MPA）中频繁的页面刷新，降低了加载延迟。

单页面应用也是一种特殊的 Web 应用程序，仅包含一个页面。它通过动态更新当前页面内容与用户交互，避免了页面切换带来的用户体验中断。在 SPA 中，所有必要的代码（HTML、JavaScript 和 CSS）通过一次加载获取，或者根据用户操作动态加载，确保在任何时刻都不会重新加载页面，也不会跳转到其他页面。我们可以形象地将 SPA 视作一个杯子，早上用杯子装牛奶，中午用杯子装开水，晚上用杯子装茶。杯子本身始终不变，变化的只是杯子里的内容。如图 1-3 所示。

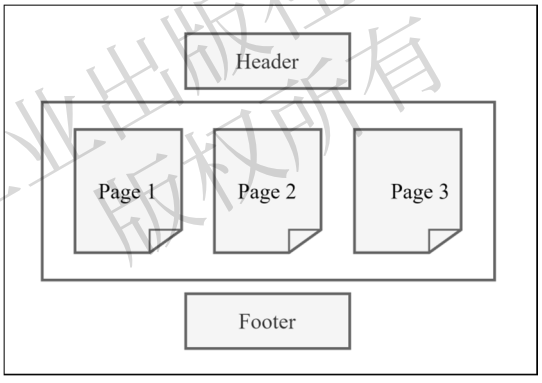

图 1-3 SPA 示意图

SPA 的优势在于其流畅的用户体验。由于页面无须重新加载，所以用户在浏览应用时能享受到更快的响应速度和更低的延迟，特别适合需要频繁交互的应用，如社交媒体平台、在线聊天工具和实时数据仪表盘。

此外，SPA 通过 AJAX 技术实现异步数据加载，用户操作时无须等待页面刷新，这使得 SPA 在构建动态内容丰富的应用时更具优势。例如，电子商务网站的产品展示、用户评论和购物车功能就可以通过 SPA 进行无缝更新。

然而，SPA 也有局限性，尤其是在 SEO 方面。由于 SPA 中的内容主要通过 JavaScript 动态生成，所以搜索引擎爬虫可能无法有效抓取页面内容。对于需要良好搜索引擎优化的内容网站，SPA 需要通过服务端渲染等技术进行优化。

MPA 与 SPA 不同。在 MPA 中，每个页面都是独立的，在访问新页面时需要重新加载 HTML、CSS 和 JavaScript 文件，公共文件则按需加载，如图 1-4 所示。

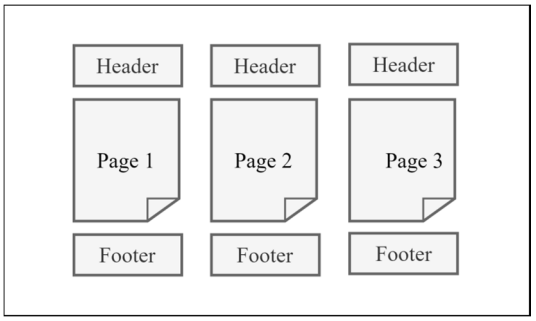

图 1-4　MPA 示意图

MPA 更适合内容丰富、结构复杂的网站，如新闻网站、企业官网和在线文档管理系统。这些网站通常包含大量不同的页面，每个页面都有独特的信息和功能。

MPA 的优势在于良好的 SEO 友好性。MPA 的每个页面都有独立的 URL，便于搜索引擎索引和排名。此外，MPA 的首次加载速度通常较快，这是因为每个页面在请求时会加载其所需的所有资源。MPA 的缺点是页面跳转可能影响用户体验，尤其是在网络环境不佳时，加载时间会延长。

综上所述，SPA 和 MPA 各有优劣。在选择架构时，应根据项目的具体需求、用户体验要求和 SEO 要求来综合评估。对于需要快速交互和动态内容的应用来说，SPA 是理想选择；而对于内容丰富且需要良好搜索引擎优化的网站来说，MPA 则更为合适。

习题 1.1

一、单选题

1.（　　）被称为“万维网之父”。

A. 比尔·盖茨　　B. 蒂姆·伯纳斯–李

C. 史蒂夫·乔布斯　　D. 谷歌创始人

2.（　　）技术允许后端动态生成网页。

A. HTML5　　B. AJAX　　C. CGI　　D. CSS

3. 首款商业浏览器 Navigator 是由（　　）公司发布的。

A. 微软　　B. 苹果　　C. 谷歌　　D. 网景

4.（　　）技术标志着 Web 2.0 时代的到来。

A. AJAX　　B. ECMAScript 6　　C. Node.js　　D. PHP

5.（　　）JavaScript 引擎的诞生极大提升了浏览器的性能。

A. Chakra　　B. SpiderMonkey　　C. V8　　D. Nitro

6. 在前端技术发展中，（　　）事件标志着 JavaScript 的标准化和全球化。

A. JavaScript 的诞生

B. 微软推出 IE 浏览器

C. W3C 的成立

D. JavaScript 提交到 ECMA 组织成为 ECMAScript

7. 在 MVC 架构模式中，（　　）组件负责接收用户输入和数据展示。

A. Model（模型）　　B. View（视图）

C. Controller（控制器）　　D. ViewModel（视图模型）

8. 在 MVVM 架构模式中，（　　）组件负责 View 和 Model 之间的数据传递。

A. Model　　B. View

C. ViewModel　　D. Controller

9. MVVM 架构模式的核心特点是（　　）。

A. 单向数据流　　B. 双向数据绑定

C. 只处理后端逻辑　　D. 通过 Controller 连接 Model 和 View

10. 单页面应用（SPA）的主要缺点是（　　）。

A. SEO 不友好　　B. 首次加载速度慢

C. 用户体验不佳　　D. 需要频繁的页面跳转

二、判断题

1. Web 1.0 时代，网页的主要特点是静态展示，没有动态内容。（　　）

2. HTML5 的发布标志着 Web 技术正式迈入了成熟阶段，支持更多多媒体内容的交互。（　　）

3. 由于单页面应用（SPA）的内容主要通过 JavaScript 动态生成，因此在 SEO 方面存在天然劣势。（　　）

任务 1.1　前端框架技术调研

构建面向未来的技术选型体系，制定符合工程化标准的前端解决方案，设计一个可实现的个人前端项目。具体任务要求如下。

1. 盘点技能

（1）回顾并总结已学的前端技术和开发经验。

（2）明确列出熟练掌握的前端技能和工具。

2. 掌握趋势

（1）调研当前前端技术的最新趋势。

（2）识别并学习新兴的前端技术和方法。

3. 设计架构

（1）基于调研结果，构思一个个人前端项目。

（2）规划项目所采用的技术栈，包括但不限于框架、库和工具等。

撰写一份综合报告，内容涵盖以上三个任务要求。

通过本任务实践，读者将掌握企业级技术选型方法论，具备架构演进规划能力，建立技术风险评估体系，形成技术决策文档化习惯。

1.2 前端开发工具

前端开发工具在日常编码中起着不可或缺的作用，对于开发者来说，选择一款好用的开发工具尤为重要。当前市场上的前端开发工具众多。其中 Visual Studio Code（简称 VS Code）以其轻量级、高性能、丰富的扩展生态和强大的功能，成为许多前端开发者的首选工具。无论是新手，还是经验丰富的开发者，VS Code 都能提供良好的开发体验和高效的工作流程。因此，本书采用 VS Code 作为前端开发工具。

1.2.1 VS Code 简介与安装

VS Code 是一款由微软开发的跨平台免费源代码编辑器，支持 Windows、Linux 等多个操作系统。VS Code 默认支持多种编程语言，包括 HTML、CSS、JavaScript 和 TypeScript，用户还可以通过下载扩展来支持 Python、C/C++、Java 和 Go 等其他语言。

该软件具有语法高亮、代码自动补全、代码重构和查看定义等功能，并内置命令行工具和 Git 版本控制系统。用户可以通过更改主题和键盘快捷方式来实现个性化设置，还可以通过内置的扩展商店安装各种扩展来拓展软件功能。

VS Code 的安装过程非常简单：首先，访问 VS Code 官网，根据所使用的操作系统下载对应的安装文件；接着，双击下载的.exe 安装文件，按照提示完成安装即可。

1.2.2 VS Code 界面介绍

VS Code 的界面设计简洁直观，旨在为开发者提供高效的编程体验，其主要界面组件如图 1-5 所示。

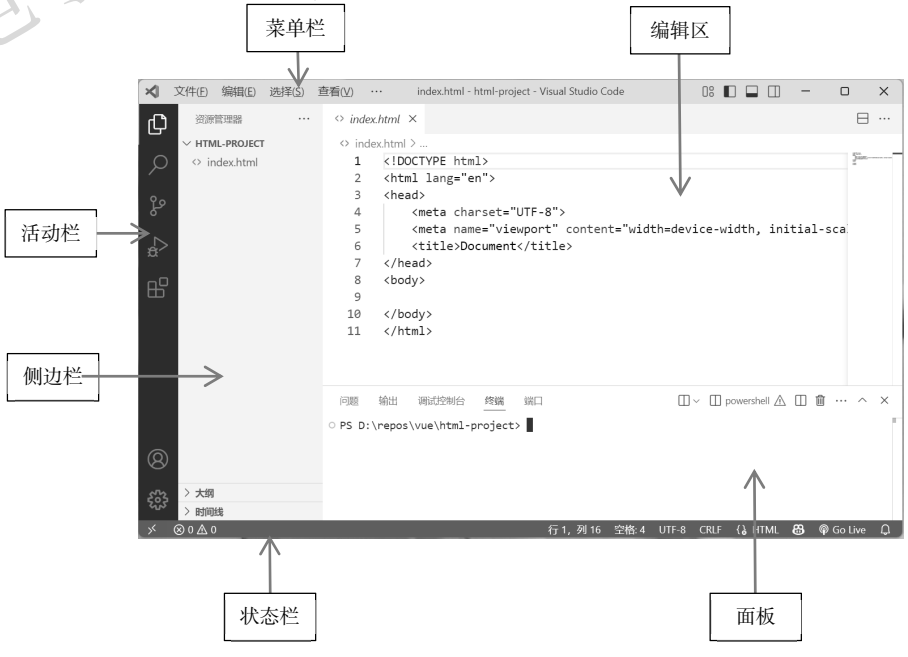

图 1-5 VS Code 主要界面组件

1. 菜单栏

菜单栏位于界面顶部，包括文件、编辑、选择、查看、转到、运行、终端、帮助等多个菜单选项，提供大部分功能。

2. 活动栏

活动栏位于界面左侧，用于访问不同的功能模块，默认情况下有七个图标，分别是资源管理器、搜索、源代码管理、运行和调试、扩展、账户及管理。

3. 侧边栏

侧边栏主要是资源管理器，显示当前项目的文件结构，用户可以在此浏览、打开和管理项目文件。右击文件或文件夹可以访问多种操作选项，如新建、删除和重命名。

4. 编辑区

界面中间的编辑区用于编写和编辑代码。VS Code 支持多标签页，用户可以同时打开多个文件，方便进行对比和编辑。

5. 状态栏

状态栏位于界面底部，显示当前项目的状态信息，如当前文件类型、Git 状态、编码格式和行号等。用户还可以通过状态栏访问一些常用功能，如更改文件编码和语言模式。

6. 面板

面板中显示的是 VS Code 内置的终端，用户可以在面板中直接运行命令，无须切换到外部终端。这些内置终端支持多种命令行工具，方便用户进行构建、测试和其他操作。

总体而言，VS Code 的界面设计旨在提高开发效率，所有功能模块都易于访问且操作简便。无论是新手，还是经验丰富的开发者，VS Code 的界面都能提供良好的用户体验，帮助用户专注于编写优质代码。

1.2.3 VS Code 常用插件

VS Code 是一个功能强大的代码编辑器，它通过丰富的扩展插件生态系统，为用户提供了无限的定制可能性。这些插件可以帮助开发者提高编码效率，优化开发流程，提升开发体验。

以下是前端开发中常用的 VS Code 插件。

（1）Chinese (Simplified) Language Pack for Visual Studio Code：VS Code 默认使用的界面语言为英文，通过安装此插件可以将界面语言切换为简体中文，方便中文开发者使用。在安装该插件后需要重启 VS Code，才能使语言包生效。

（2）Vue-Official：Vue 3 官方推荐的配套插件，提供对 Vue 文件的语法高亮、代码片段、智能提示（如属性补全、自动完成、方法参数提示等）及错误检查功能，能够极大简化 Vue 开发流程。

（3）ESLint：常用的代码质量检查插件，能够帮助开发者保持代码风格一致，预防潜在错误。也可以与 Prettier 结合使用，实现代码检查与格式化的双重保障。

（4）Prettier：自动格式化代码，确保代码风格的统一和整洁。可与 ESLint 配合使用。

（5）Auto Rename Tag：在编辑大型 HTML 文件时，此插件可以自动同步修改 HTML/XML 标签的配对标签，减少手动查找和修改配对标签的工作量，提高编码效率。

（6）Path Intellisense：提供路径自动补全功能，当输入部分路径时，此插件会智能提示文件或文件夹路径，加快文件和文件夹路径的输入速度，减少手动查找和输入路径的时间和错误，提高编码效率。

（7）JavaScript (ES6) Code Snippets：提供 ES6 语法的智能提示和代码片段补全功能，帮助开发者快速编写现代 JavaScript 代码。

（8）Open in Browser：允许用户通过右键快捷菜单直接在浏览器中打开当前 HTML 文件，方便快速预览页面效果，加快前端开发流程。

（9）Live Server：为前端开发者提供本地实时预览功能。安装完成后，在 HTML 文件中单击鼠标右键，在弹出的快捷菜单中选择“Open with Live Server”命令，即可启动本地服务器并预览网页。在修改代码并保存后，浏览器会自动刷新，实时展示效果，极大提升了开发效率。它还支持多种前端技术框架，兼容性良好。

可以通过以下步骤来使用插件。

（1）安装插件：打开 VS Code，单击左侧活动栏中的“扩展”图标或按快捷键 Ctrl+Shift+X（Windows/Linux）/Cmd+Shift+X（macOS）。在文本框中输入要安装的插件名称，找到插件后单击“安装”按钮即可。

（2）配置插件：打开 VS Code 的设置界面，可以单击左下角的“管理”图标或按快捷键 Ctrl+“,”（Windows/Linux）/Cmd+“,”（macOS）打开。在文本框中输入插件名称，找到插件后即可根据个人需求进行配置。

通过以上介绍，读者应该已经了解了一些常用的 VS Code 插件，并能够根据自己的开发需求进行选择和配置。这些插件不仅能够提升编码效率，还能优化开发体验。VS Code 的扩展商店中还有更多优秀的插件等待读者探索，以便进一步定制和优化个人编码环境。

1.2.4 VS Code 集成 AI 代码工具

随着人工智能技术的快速发展，AI 代码工具逐渐成为开发者提升编码效率的重要助手。VS Code 作为一款功能强大的代码编辑器，支持通过插件集成多种 AI 代码工具。这些工具能够通过分析代码上下文、学习大量开源项目代码，帮助开发者显著提高开发效率和代码质量。

AI 代码工具在前端开发中具有广泛的应用场景，它提供了多种功能，包括代码补全、代码生成、错误检测与修复及代码优化。

- 在代码补全方面，AI 代码工具可以根据上下文自动补全代码。例如，在编写 Vue 组件时，若输入<template>，则 AI 代码工具会自动补全<script>和<style>部分；在编写 JavaScript 函数时，它可以根据函数名和参数自动补全函数体。

- 在代码生成方面，AI 代码工具可以根据注释或需求生成代码。例如，输入注释“//创建一个 Vue 组件，显示用户信息”，AI 代码工具会自动生成对应的 Vue 组件代码；输入注释“//实现一个购物车功能”，它会生成购物车相关的代码框架。
- 在错误检测与修复方面，AI 代码工具可以实时检测代码中的语法错误和逻辑错误，并提供修复建议。例如，当 Vue 组件中的 v-for 指令缺少 key 属性时，AI 代码工具会提示添加 key 属性。
- 在代码优化方面，AI 代码工具可以分析代码性能并提供优化建议。例如，当检测到重复代码时，AI 代码工具会建议将其提取为函数或组件，从而提高代码的可维护性和性能。

通过这些功能，AI 代码工具可以帮助开发者更快地编写高质量代码，减少重复劳动，使开发者专注于解决复杂的开发问题。

常见的 AI 代码工具如下。

- GitHub Copilot：由 GitHub 和 OpenAI 联合开发，支持多种编程语言，能够根据注释或代码片段生成完整的代码。
- Tabnine：基于深度学习的代码补全工具，支持多种语言和框架，能够根据上下文提供精准的代码建议。
- Codeium：一款免费的 AI 代码工具，支持代码补全、代码生成和代码优化。
- Continue：专注于代码上下文理解的 AI 代码工具，支持根据当前开发场景生成连贯的代码片段，尤其擅长复杂逻辑的实现。
- Cline：轻量级 AI 代码助手，专注于 JavaScript/TypeScript 生态，提供智能代码补全和实时错误检测功能。

下面以 GitHub Copilot 和 Continue 为例，介绍如何在 VS Code 中集成 AI 代码工具。

1. GitHub Copilot 的安装和使用

安装 GitHub Copilot 插件的步骤为：打开 VS Code，单击“扩展”图标，在打开页面的搜索框中输入“GitHub Copilot”，找到插件后单击“安装”按钮。安装完成后，VS Code 会提示登录 GitHub 账号以激活 Copilot。

在登录 GitHub 账号后，Copilot 会自动启用。用户可以选择不同的 AI 模型，如 Claude 和 GPT 等。也可以在编写代码时，在代码文件中按快捷键 Ctrl+I，打开选择 AI 模型对话框，如图 1-6 所示，在该对话框中选择“GPT-4.1”选项。选择 AI 模型后，在输入框中输入提示词并按下 Enter 键，Copilot 将调用所选 AI 模型提供代码建议。可以单击“接受”按钮接受建议，也可以单击“关闭”按钮拒绝建议。

2. Continue 结合 DeepSeek 的安装和使用

DeepSeek 是一款高效且功能强大的 AI 模型。与传统大模型相比，DeepSeek 具有低算力需求优势，适合在资源受限的环境中使用。此外，它还支持多模态交互，具备实时数据处理能力，可广泛应用于代码开发、数据分析、智能客服等多个场景。DeepSeek 的简洁易用和高效性能，使其成为开发者在大数据时代抢占先机的重要工具。用户可以在 VS Code 编辑器中安装 Continue 插件来引入 DeepSeek，即可通过 DeepSeek 助力前端开发。

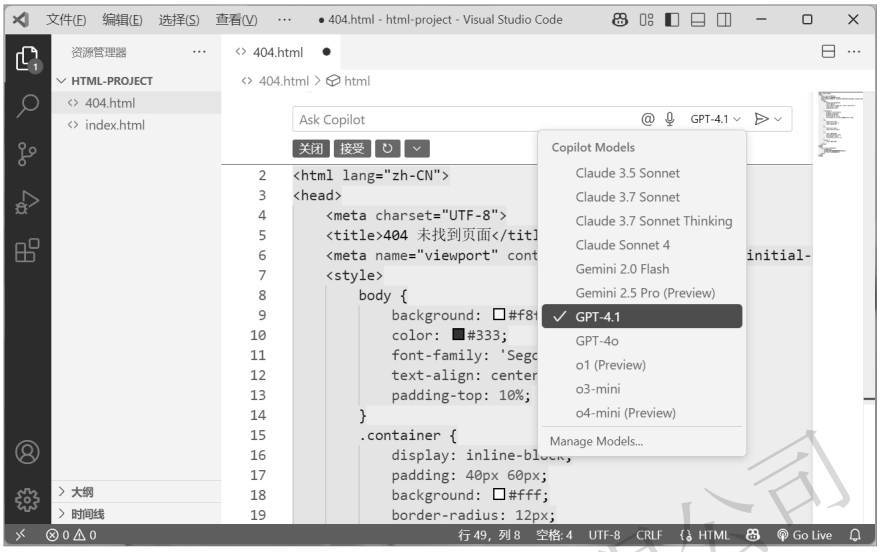

图 1-6　选择 AI 模型对话框

安装 Continue 插件的方法与安装 GitHub Copilot 插件的方法相似。在安装完成后，左侧活动栏中会多出 Continue 的图标，单击该图标即可打开“CONTINUE”面板。

在“CONTINUE”面板中展开模型下拉列表，选择“Add Chat model”选项，如图 1-7 所示。打开“Add Chat model”对话框，在“Provider”下拉列表中选择“DeepSeek”选项，在“Model”下拉列表中选择“DeepSeek Coder”选项，在“API key”文本框中输入读者的 DeepSeek API key，如图 1-8 所示。

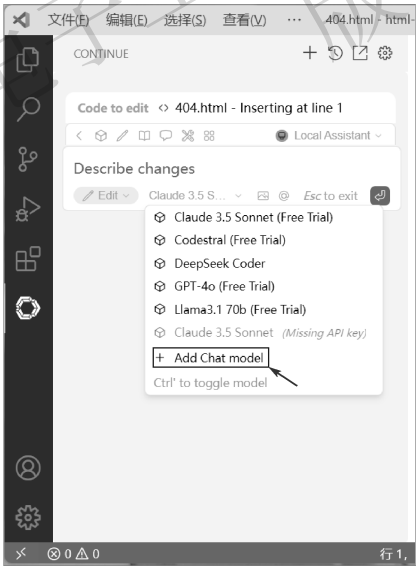

图 1-7　选择“Add Chat model”选项

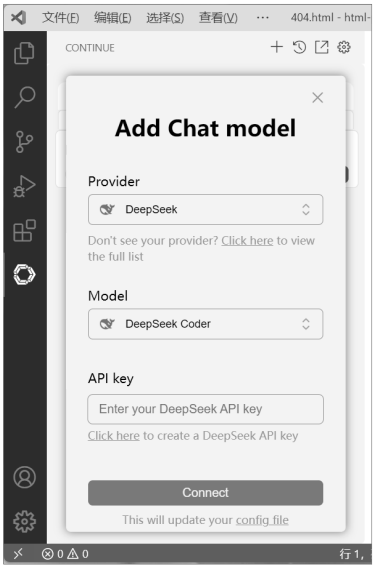

图 1-8　“Add Chat model”对话框

DeepSeek API key 需要在“deepseek 开放平台”中获取。读者在登录“deepseek 开放平台”后，先单击左侧列表框中的“API keys”节点，再单击“创建 API key”按钮，即可获取 DeepSeek API key，如图 1-9 所示。

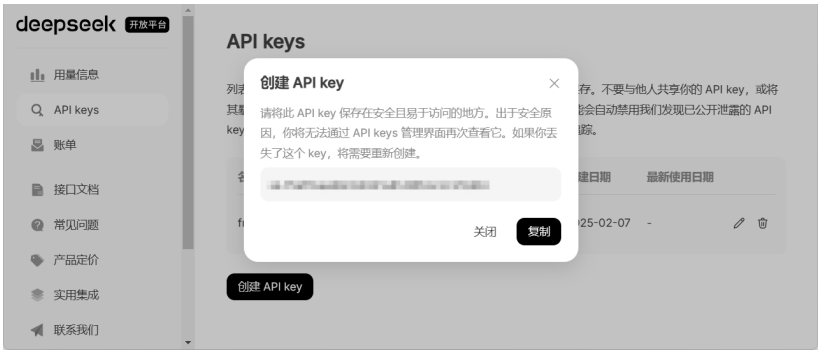

图 1-9　在 DeepSeek 开放平台创建 DeepSeek API key

完成以上配置后，读者就可以在“CONTINUE”面板的输入框中输入需求，在按下 Enter 键后，Continue 将调用 DeepSeek 提供代码建议，如图 1-10 所示，接着单击“插入代码”“复制代码”等按钮进行下一步操作。此外，读者也可以在代码文件中按快捷键 Ctrl+I，快速唤起“CONTINUE”面板。

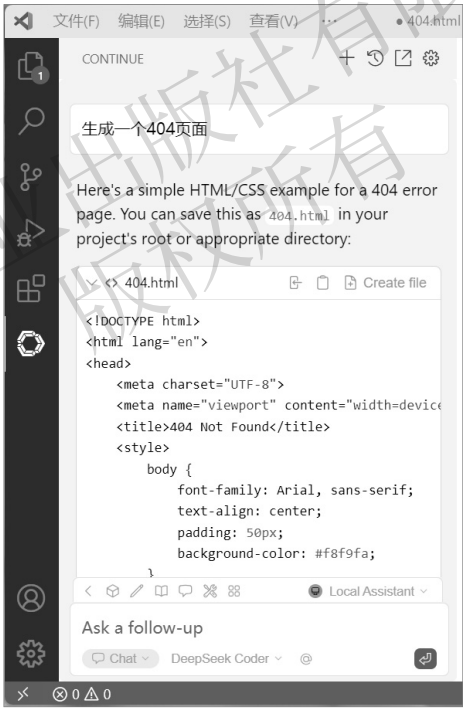

图 1-10　Continue 调用 DeepSeek 提供代码建议

通过集成 AI 代码工具，开发者可以显著提升编码效率，减少重复劳动，并专注于业务逻辑的实现。VS Code 提供了丰富的插件支持，使得 AI 代码工具能够无缝融入开发流程。在实际项目中，合理使用 AI 代码工具，结合人工检查与优化，能够帮助开发者更快、更好地完成前端开发任务。当然，AI 代码工具还有更多功能，等待读者自行探索。

不过，在使用 AI 代码工具时，开发者需注意多方面事项。一方面，AI 生成的代码可能不完全符合项目规范，这就要求开发者仔细检查并进行调整，以确保代码质量达到项目要求。

另一方面，隐私与安全问题至关重要，在使用过程中要避免将敏感信息，如 API 密钥等，写入代码注释或提示中，防止信息泄露。此外，AI 代码工具的使用并非毫无门槛，它需要开发者投入一定的时间和精力去学习，熟悉其功能与配置，才能更好地发挥工具的作用。

通过本节的学习，读者将掌握如何在 VS Code 中集成 AI 代码工具，并能够利用这些工具提升前端开发效率。在后续的任务中，读者将通过实际项目练习，进一步熟悉 AI 代码工具的使用技巧。

1.2.5 VS Code 常用快捷键

快捷键是每个编辑器的重要组成部分，它能显著提升开发效率。因此，熟练掌握 VS Code 的快捷键可以提高编码速度和效率。

图 1-11　选择“键盘快捷方式”选项

VS Code 预置了大量快捷键，用户可以直接使用，也可以根据个人习惯在快捷键设置中自定义绑定特定操作。若要更改快捷键，则可以单击活动栏中的“管理”图标，在打开的页面中选择“键盘快捷方式”选项，如图 1-11 所示。在打开快捷键列表后，双击想要修改的快捷键即可进行自定义设置。

以下是一些常用的快捷键。

1. 多光标操作

当需要同时编辑多个位置时，可以按住 Alt 键并使用鼠标左键单击多个位置，或者按快捷键 Alt+Ctrl 并使用上下箭头来新增光标，实现多光标操作。

2. 复制一行/多行代码

如果要快速复制一行或多行代码，那么首先要选中内容（复制一行时，光标放在行内即可），然后按快捷键 Alt+Shift 并使用上下箭头来复制行，将其插入目标位置。

3. 移动行/代码块

当需要上下移动某一行或选中的代码块时，按住 Alt 键并使用上下箭头即可轻松移动。

更多的快捷键操作可以查阅快捷键列表。通过合理运用这些快捷键，能够显著提高开发效率并减少重复操作。用户还可以根据自己的需求进一步自定义快捷键，打造更顺手的开发体验。

习题 1.2

一、单选题

1. Visual Studio Code（VS Code）是由（　　）公司开发的。

A. 谷歌　　B. 网景　　C. 苹果　　D. 微软

2. VS Code 默认支持（　　）编程语言。

A. HTML、CSS、JavaScript 和 TypeScript　　B. Python、C/C++、Java

C. Go、Ruby、Perl　　D. PHP、Swift、Kotlin

3.（　　）功能是 VS Code 的内置功能。

A. 视频编辑　　B. Git 版本控制　　C. 图片编辑　　D. 数据库管理

4. VS Code 的（　　）组件位于界面左侧，包含多个图标用于访问不同的功能模块。

A. 菜单栏　　B. 活动栏　　C. 主侧边栏　　D. 状态栏

5. VS Code 内置的扩展商店允许用户安装（　　）类型的扩展。

A. 主题和图标包　　B. 代码片段和动画效果

C. 功能性扩展和语言支持　　D. 游戏和娱乐应用

6. 在 VS Code 中，（　　）插件可以帮助开发者检查代码风格和避免常见错误。

A. ESLint　　B. Path Intellisense

C. Vue-Official　　D. Auto Rename Tag

7. 在 VS Code 中，（　　）插件提供对 Vue 文件的语法高亮和智能提示功能。

A. ESLint　　B. Vue-Official

C. Auto Rename Tag　　D. JavaScript (ES6) Code Snippets

8. 在 VS Code 中，（　　）用于编写和编辑代码。

A. 菜单栏　　B. 活动栏　　C. 编辑区　　D. 状态栏

9. 在 VS Code 中，快速在浏览器中打开当前 HTML 文件的方法是（　　）。

A. 使用 Open in Browser 插件

B. 使用快捷键 Ctrl+P

C. 在菜单栏中选择“文件”→“打开”命令

D. 将文件拖动到浏览器窗口

10. 在 VS Code 中，（　　）插件是由 GitHub 和 OpenAI 联合开发的，支持多种编程语言，能够根据注释或代码片段生成完整的代码。

A. Tabnine　　B. GitHub Copilot　　C. Codeium　　D. Continue

11. 在 VS Code 中，打开键盘快捷键设置界面的方法是（　　）。

A. 在菜单栏中选择“查看”→“命令面板”命令

B. 在菜单栏中选择“帮助”→“快捷键参考”命令

C. 按快捷键 Ctrl+Shift+P

D. 单击活动栏中的“管理”图标，接着选择“键盘快捷方式”选项

二、判断题

1. Visual Studio Code 是一款由微软开发的跨平台免费源代码编辑器，支持 Windows、Linux 等多个操作系统。（　　）

2. VS Code 默认支持多种编程语言，包括 HTML、CSS、JavaScript 和 TypeScript，但不支持 Python、C/C++、Java 和 Go 等其他语言。（　　）

3. VS Code 中的 Chinese (Simplified) Language Pack 插件可以将界面语言切换为简体中

文，且安装后无须重启 VS Code 即可生效。（　　）

4. VS Code 的快捷键 Alt + Shift +上下箭头可以用来复制选中的行或代码块。（　　）

任务 1.2　掌握 VS Code 高效开发技巧

通过系统化学习与实践，熟悉并掌握使用 VS Code 进行高效开发的技巧。具体任务要求如下。

1. 环境搭建

（1）下载并安装 VS Code 编辑器，完成基础配置。

（2）熟悉 VS Code 用户界面，包括菜单栏、活动栏、编辑区、面板等核心组件。

2. 效率提升

（1）掌握常用快捷键，通过反复练习形成肌肉记忆。

（2）根据前端开发需求，安装常用插件，如 Vue-Official、ESLint、Open in Browser 等。

3. 实战验证

（1）创建一个简单的 HTML 页面，使用快捷键完成编辑、保存、预览全流程。

（2）通过插件功能（如 ESLint）检查代码规范，修正提示问题。

通过本任务实践，读者将掌握开发环境配置能力，构建智能化编码工作流体系，实现工程级调试与优化方案，建立团队协同开发规范认知。

1.3　Vue.js 入门

前端技术在互联网行业中占据着日益重要的地位，无论是 PC 端、移动端，还是智能手表等设备，都离不开前端的支持。在前端技术的发展历程中，涌现出许多优秀的框架，每个框架都有其独特的优势和适用场景。

其中，Vue.js 因其轻量级、易上手、学习曲线平缓等特点，广受前端开发者的青睐，尤其适合新手学习。与其他框架相比，Vue.js 在结构、样式和业务逻辑的分离方面更为清晰和彻底，能够极大地简化开发流程。其丰富的学习资源、完善的生态系统进一步降低了入门难度。Vue.js 在国内得到了广泛应用，拥有众多知名企业的支持，成为企业级开发的重要选择。

1.3.1　Vue.js 简介

Vue.js（简称 Vue）是一款用于构建用户界面的开源渐进式 JavaScript 框架。其“渐进式”设计理念使其能够根据项目需求灵活使用。用户可以仅使用 Vue.js 的核心功能快速开发视图层，也可以结合官方的“全家桶”工具（如 Vue Router、Vuex 等）构建复杂的单页面应用。这种灵活性让 Vue.js 能够适应从简单项目到大型应用的多样化场景。

Vue.js 的核心库专注于视图层，上手容易且便于与第三方库或现有项目集成。同时，它也能与现代化工具链和各种支持库结合，轻松驱动复杂的前端应用程序。通过组件内部的方法，

Vue.js 实现了数据与视图之间的双向绑定，简化了界面与数据的交互。

Vue.js 的主要特点如下。

（1）渐进式架构：根据项目规模和复杂度逐步引入功能，从核心功能到完整框架体系。

（2）轻量级且高性能：核心库体积小，加载速度快，适合移动端和桌面端开发。

（3）双向数据绑定：自动同步数据和视图，减少手动 DOM 操作，降低代码复杂度。

（4）组件化开发：将页面拆分为独立、可复用的组件，提升开发效率和代码可维护性。

（5）易于上手：简洁直观的 API，平缓的学习曲线，适合前端开发新手。

（6）灵活的生态系统：提供丰富的官方工具，并支持与其他库和框架无缝集成。

（7）强大的社区支持：活跃的社区和丰富的教程、插件，帮助开发者快速解决问题并提高开发效率。

凭借这些优势，Vue.js 成为前端开发领域备受欢迎的框架之一，广泛应用于各类项目。

1.3.2 Vue.js 脚本引入

Vue.js 采用渐进式设计理念，允许开发者根据项目需求灵活选择集成方案。常见方式有两种：脚本直接引入和工程化构建。为帮助读者建立渐进式学习路径，本书前三章采用脚本直接引入方式实现快速实践，后续章节将过渡到工程化构建方式。这种分层学习策略，既能指导初学者通过脚本直接引入快速建立框架认知，又能引导进阶者通过工程化构建深入理解现代开发流程。

脚本直接引入有以下三种方式。

1. 动态版本引入

通过<script>标签加载云端资源是最快捷的集成方式。推荐采用内容分发网络（CDN）获取官方资源包，这种方案既能保证加载速度，又能自动获取最新稳定版本。目前有多种 CDN 服务可供选择，unpkg 是一个常用的 CDN 服务，它直接从 NPM 获取资源，提供了快速且全球分布的 CDN 网络。此外，读者也可以选择其他 CDN 服务，如 jsDelivr 或 cdnjs 等。

读者可以根据 unpkg 和 vue 这两个关键字，先在网络上搜索得到 Vue.js 脚本的托管地址，再通过以下代码引入到项目中。

```
<script src="unpkg 网址/vue@3"></script>
```

需要注意的是，src 属性中 Vue.js 脚本的完整引用地址需读者根据实际搜索结果进行完善。其中，@3 表示锁定主版本号为 Vue 3，云端服务会自动解析为当前最新子版本。该方式适合体验最新特性，但不同子版本之间可能存在细微差异。

2. 静态版本锁定

生产环境推荐使用精确版本控制策略，通过完整版本号确保开发环境与部署环境的一致性。版本号格式通常为“主版本.次版本.修订号”，示例如下。

```
<script src="unpkg 网址/vue@3.5.13"></script>
```

将占位符 x.x.x 替换为具体版本数字（如上例中的 3.5.13），既可享受版本稳定性保障，又

能通过人工控制实现渐进升级。这种方案能有效避免因自动升级而导致的兼容性问题，是团队协作开发的首选方式。实际集成时应通过包管理工具或官方文档获取最新版本号，此处版本号仅为格式示意。

3. 本地自行托管

若网络环境受限，则可以下载构建文件并托管至本地（如 static/js 目录），此方案需手动维护文件更新，但能完全掌控依赖来源，示例如下。

```
<script src="./static/js/vue.global.js"></script>
```

在构建版本的选择上，Vue.js 提供了多种构建版本，以适应不同场景。

- 开发版（如 vue.global.js）：包含完整警告信息与调试功能，适合开发阶段。
- 生产版（如 vue.global.prod.js）：移除调试代码并压缩，体积更小，专为部署优化。

图 1-12 所示为 Vue.js 3.4.38 版本中 dist 目录下的构建版本。

开发者可根据实际环境，通过构建工具或 CDN 路径切换版本（见图 1-12）。当项目复杂度提升时，建议通过工程化工具自动处理构建流程，实现开发效率与代码质量的双重保障。

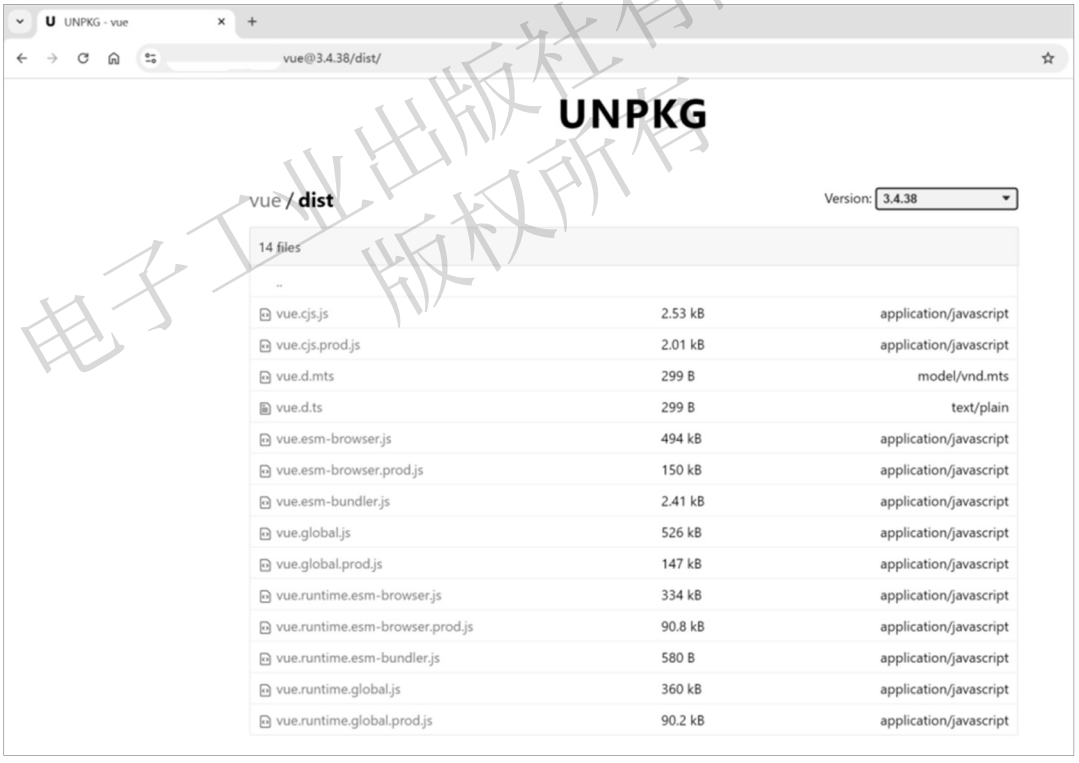

图 1-12　Vue.js 3.4.38 版本中 dist 目录下的构建版本

1.3.3　Vue.js 应用实例创建

新建项目文件夹 html-project，并在 VS Code 中打开，在项目根目录下新建一个名为 hello.html 的文件，实现第一个 Vue.js 页面，代码如下。

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
    <script src="unpkg 网址/vue@3"></script>
  </head>
  <body>
    <div id="app">
      <h1>{{ title }}</h1>
    </div>
    <script>
      const App = {
        data() {
          return {
            title: "Hello Vue.js",
          };
        },
      };
      Vue.createApp(App).mount("#app");
    </script>
  </body>
</html>
```


在以上代码中，主要有以下几个要点。

（1）在页面中通过<script>标签导入 Vue.js 文件。

（2）在<body>中创建一个 id 为 app 的 div 元素，用于初始化 Vue。

（3）在 JavaScript 代码中，通过全局暴露出来的 Vue 的 createApp()方法创建一个 Vue.js 应用实例（以下简称 Vue 应用实例），并将该应用实例通过 mount()方法挂载到 id 为 app 的元素上。

代码编写完成后，直接用浏览器打开这个文件，就可以看到运行效果，页面中会显示 Hello Vue.js，如图 1-13 所示。

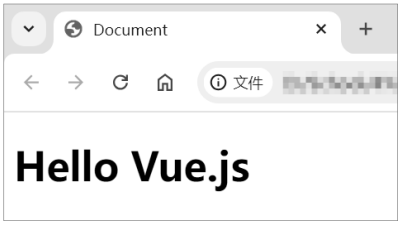

图 1-13 第一个 Vue.js 页面的运行效果

习题 1.3

一、单选题

1. Vue.js 的核心库主要关注（　　）的开发。

A. 模型层　　B. 视图层　　C. 控制层　　D. 数据层

2. Vue.js 被称为“渐进式”框架的原因是（　　）。

A. 它可以根据项目需求逐步引入更多功能

B. 它只能用于小型项目
C. 它的性能逐渐提升
D. 它的功能都集成在核心库中

3. Vue.js 的（　　）特性极大地减少了手动操作 DOM 的需求。

A. 组件化开发　　B. 语法高亮　　C. 双向数据绑定　　D. 状态管理

4. Vue.js 是通过（　　）机制实现数据和视图的自动同步的。

A. 单向数据流　　B. 双向数据绑定　　C. 手动 DOM 操作　　D. 服务器端渲染

5. 在渐进式引入 Vue.js 时，通常推荐通过（　　）方式导入 Vue.js 文件。

A. 使用 CDN　　B. 使用 Nginx
C. 手动编写 JavaScript 文件　　D. 通过 Node.js 安装

6. 在 Vue.js 中，被用来渐进式引入 Vue.js 的 CDN 服务是（　　）。

A. unpkg　　B. jsDelivr　　C. cdnjs　　D. 以上都是

7. 在开发环境中，建议导入 Vue.js 的版本号文件是（　　）。

A. vue.global.js　　B. vue.global.prod.js　　C. vue.global.dev.js　　D. vue.global.min.js

8. 在 Vue.js 中，创建应用实例的常用方法是（　　）。

A. Vue.initApp()　　B. Vue.buildApp()　　C. Vue.startApp()　　D. Vue.createApp()

9. 要在 Vue.js 中挂载应用实例到页面的某个元素上，使用的函数是（　　）。

A. mount()　　B. bind()　　C. attach()　　D. link()

10. Vue 应用实例的数据对象通常定义在（　　）中。

A. props　　B. model　　C. data　　D. methods

二、判断题

1. Vue.js 是一款用于构建用户界面的开源渐进式 JavaScript 框架。（　　）
2. Vue.js 的核心库专注于视图层，因此它无法与第三方库或现有项目进行集成。（　　）
3. Vue.js 支持双向数据绑定，这意味着数据和视图之间可以自动同步。（　　）
4. Vue.js 的安装可以通过 CDN 服务引入，也可以下载 JavaScript 文件并自行托管。（　　）
5. Vue.js 的安装只能通过脚手架工具进行。（　　）

任务 1.3　开发个人主页

通过开发“HTML 版技术社区网站”的个人主页，认识 Vue.js 核心概念（如应用实例化、数据绑定），理解渐进式开发模式，并尝试运用 AI 代码工具辅助优化代码质量与页面表现。具体任务要求如下。

1. 环境搭建与基础配置

新建 HTML 版技术社区网站的项目文件夹 tech，按以下结构初始化资源。

```
tech/
├─── static/
|    ├─── js/                # 存放 vue.global.js 文件
|    ├─── css/               # 存放后续样式文件
|    └─── images/            # 存放图片资源
└─── userProfile.html       # 个人主页
```

从官方 CDN 下载 Vue.js 脚本（如 vue.global.js），并将其存入 static/js 目录。

2. Vue 应用实例创建与数据绑定

在 userProfile.html 中导入 Vue.js 脚本，创建名为 app 的 Vue 应用实例，并定义数据对象，代码如下。

```
            user: {
              name: "张三",
              age: 28,
              gender: "男",
              hobby: "阅读、运动",
              job: "前端开发工程师",
              bio: "热爱学习新技术，喜欢分享和交流。",
            },
```

使用{{ }}插值语法，在页面中动态展示用户信息。

3. 页面设计与 AI 优化

编写页面代码，实现基础样式，展示用户信息，个人主页展示页面效果如图 1-14 所示。

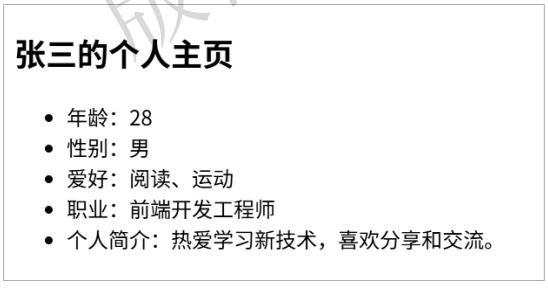

图 1-14 个人主页展示页面效果

打开 AI 代码工具面板，输入提示词，生成美化代码，选择性采纳 AI 建议，更新页面样式（如阴影效果、字体排版）。

4. 功能验证与提交

确保页面数据能够动态渲染，浏览器控制台无报错。

通过 Live Server 插件实时预览，调整布局至符合设计稿。

通过本任务实践，读者将具备 Vue 应用架构基本设计能力，实现 AI 增强型开发工作流，形成可持续演进架构思维。

1.4 前端 UI 框架入门

在前端开发中，UI 框架的出现极大地简化了界面样式的设计和实现，使开发过程更加高效。Element 和 Element Plus 是两款由饿了么前端团队开源维护的优秀 UI 框架，它们为开发者提供了丰富的组件库，能够帮助开发者快速构建美观且功能完备的页面。这些框架不仅节省了编写样式的时间，还确保了页面布局的合理性和风格的统一性，即使是没有专业设计师的团队，也能轻松实现高质量的页面设计。

Element 是基于 Vue 2.x 的 UI 框架，而 Element Plus 是其升级版本，专为 Vue 3.x 设计，它采用 TypeScript 和组合式 API 构建，充分利用了 Vue 3 的新特性。Element Plus 的官网提供了详尽的开发文档和示例代码，便于开发者快速上手。其组件库涵盖基础、布局、表单、数据展示、导航和消息提示六大类功能，可以满足大多数 PC 端项目的开发需求。此外，Element Plus 还支持自定义主题，进一步提升了灵活性。

因为本书基于 Vue 3.x，所以我们将采用 Element Plus 来介绍前端 UI 框架的使用方法。通过学习 Element Plus，读者可以更好地理解如何在 Vue 项目中集成和使用 UI 组件库，从而提升开发效率。

1.4.1 Element Plus 集成

我们可以从 Element Plus 中文官网上获取其最新版本。打开中文官网，单击顶部菜单栏中的“指南”菜单，进入指南页面，在左侧列表框中选择“安装”选项，进入“安装”页面，查看引入方式。

Element Plus 提供两种引入方式：一种是浏览器直接引入，即通过 HTML 标签直接在浏览器中引入 Element Plus，之后即可使用全局变量 Element Plus；另一种是使用包管理器，如 NPM、Yarn 或 pnpm 等，具体操作将在后续章节中介绍。

这里我们采用浏览器直接引入的方式。首先打开 html-project 项目中的第一个 Vue.js 页面，即 hello.html 文件，然后在中文官网安装页面中找到相关代码，将其复制到该 hello.html 文件的<head>标签中，即可将 Element Plus 引入该页面，代码如下。

```
    <!-- 引入脚本，注意要在 Vue.js 脚本之后引入 -->
<script src="unpkg 网址/element-plus"></script>
    <!-- 引入样式 -->
    <link rel="stylesheet" href="unpkg 网址/element-plus/dist/index.css" />
```

与 Vue.js 脚本引入一样，我们也可以对 Element Plus 进行本地自行托管。在 html-project 项目中，新建目录 static/element-plus/，下载 Element Plus 的脚本文件和样式文件，并放到该目录下，项目目录结构如图 1-15 所示。

这样就可以在页面的<head>标签中引入本地自行托管的资源，代码如下。

```
<script src="./static/element-plus/index.full.js"></script>
    <link rel="stylesheet" href="./static/element-plus/index.css" />
```

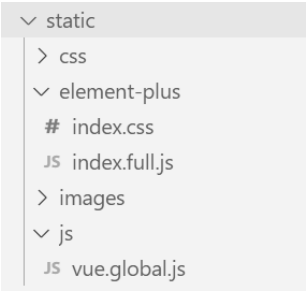

图 1-15 项目目录结构

通过以上方式成功引入 Element Plus 后，还需要通过 use()方法将 Element Plus 集成到 Vue 应用实例中，代码如下。

```
    <script>
      const App = {
        data() {
          return {};
        },
      };
      Vue.createApp(App).use(ElementPlus).mount("#app");
</script>
```

在 Vue 3 中，use()是应用实例的一个方法，用于集成插件。它的核心作用是将第三方库或自定义功能全局集成到 Vue 应用实例中。

完成以上步骤就可以使用 Element Plus 了。进入 Element Plus 中文官网，单击顶部菜单栏中的“组件”菜单，打开组件页面，如图 1-16 所示，我们可以任意选择组件来观察效果。

图 1-16 Element Plus 中文官网的组件页面

在左侧列表框中找到“Basic 基础组件”中的“Button 按钮”，复制“Primary”按钮的代码，并粘贴到文件中，代码如下。

```
<el-button type="primary">Primary</el-button>
```

打开页面，可以看到该按钮成功出现在页面上，如图 1-17 所示。

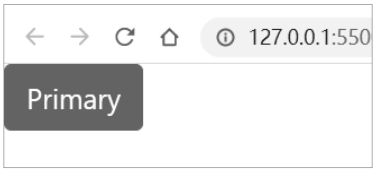

图 1-17　按钮成功出现在页面上

1.4.2　Element Plus 组件

Element Plus 有丰富的组件供前端开发者使用，包含基础组件、配置组件、表单组件、数据展示组件、导航组件、反馈组件和其他组件七大类。下面我们选取其中个别组件，通过几个简单的示例，演示组件的使用方法。

1. 基础组件

基础组件有按钮、边框、色彩、布局容器、图标、布局、链接、文本、滚动条、间距、排版等，在前端开发中都是比较常用的。

开发一个网页的第一步通常是对页面进行整体布局，此时可以使用布局容器组件。用于布局的容器组件有外层容器<el-container>、顶栏容器<el-header>、侧边栏容器<el-aside>、主要区域容器<el-main>和底栏容器<el-footer>，它们能够帮助开发者快速搭建页面的基本结构。需要注意的是，<el-container>的直接子元素必须是后四个组件中的一个或多个，后四个组件的父元素必须是<el-container>。

Element Plus 中文官网的布局容器组件页面展示了一些常见页面布局，以第二个页面布局为例，该页面布局包含 Header、Main、Footer 三大部分，如图 1-18 所示。

图 1-18　包含 Header、Main、Footer 三大部分的常见页面布局

在 html-project 项目根目录下新建 index.html 文件，通过使用布局容器组件，实现页面基本布局，代码如下。

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>首页</title>
    <script src="./static/js/vue.global.js"></script>
    <script src="./static/element-plus/index.full.js"></script>
    <link rel="stylesheet" href="./static/element-plus/index.css" />
    <style>
      .el-header,
      .el-footer,
      .el-main {
        display: flex;
        align-items: center;
        justify-content: center;
      }
      .el-header,
      .el-footer {
        background-color: #c9e1fe;
        height: 60px;
      }
      .el-main {
        background-color: #edf5ff;
        height: 150px;
      }
    </style>
  </head>
  <body>
    <div id="app">
      <el-container>
        <el-header>Header</el-header>
        <el-main>Main</el-main>
        <el-footer>Footer</el-footer>
      </el-container>
    </div>
    <script>
      const App = {
        data() {
          return {
            title: "",
          };
        },
      };
      Vue.createApp(App).use(ElementPlus).mount("#app");
    </script>
  </body>
</html>
```

2. 导航组件

Element Plus 提供了多种类型的导航组件，帮助开发者根据实际需求快速构建导航栏，实现高效布局。在 Element Plus 中文官网组件页面左侧的列表框中，找到“Navigation 导航”组件，该组件包含多种具有不同导航栏样式的组件。在这些组件中，菜单组件是常用的导航栏组件之一，如图 1-19 所示。

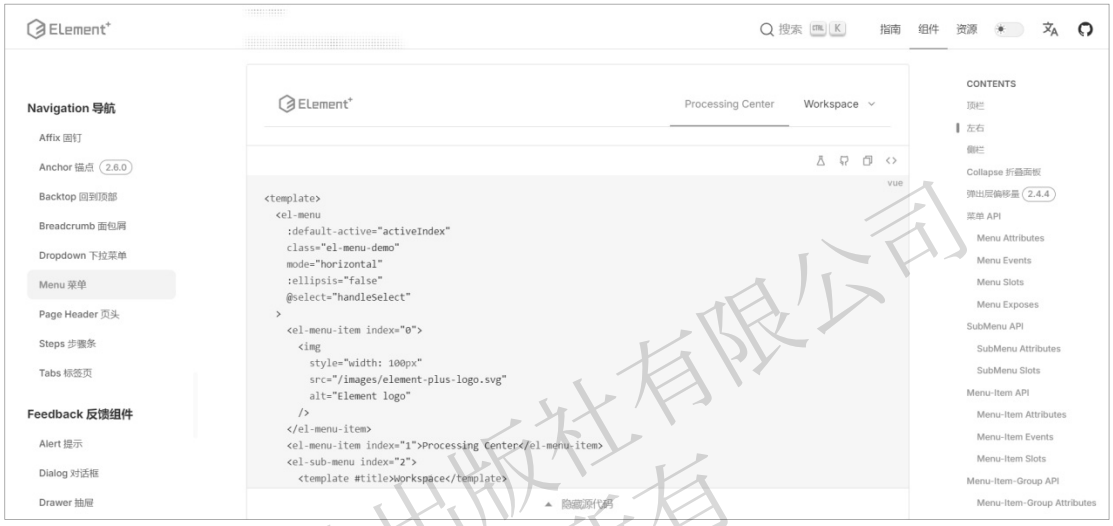

图 1-19　导航组件中的菜单组件

复制菜单组件的代码，将其粘贴到 index.html 文件中，并根据项目需要进行相应的修改，以下为修改后的代码。

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>首页</title>
    <script src="./static/js/vue.global.js"></script>
    <script src="./static/element-plus/index.full.js"></script>
    <link rel="stylesheet" href="./static/element-plus/index.css" />
    <style>
      .el-header {
        display: flex;
        justify-content: space-between;
      }
    </style>
  </head>
  <body>
    <div id="app">
      <el-container>
        <el-header>
```

```
        <div>
          <h1>{{title}}</h1>
        </div>
        <div>
          <el-menu mode="horizontal" :ellipsis="false">
            <el-menu-item index="0">菜单一</el-menu-item>
            <el-menu-item index="1">菜单二</el-menu-item>
            <el-menu-item index="2">菜单三</el-menu-item>
            <el-menu-item index="3">菜单四</el-menu-item>
          </el-menu>
        </div>
      </el-header>
      <el-main>Main</el-main>
      <el-footer>Footer</el-footer>
    </el-container>
  </div>
  <script>
    const App = {
      data() {
        return {
          title: "Hello Vue.js",
        };
      },
    };
    Vue.createApp(App).use(ElementPlus).mount("#app");
  </script>
 </body>
</html>
```


菜单组件的页面效果如图 1-20 所示。

图 1-20 菜单组件的页面效果

3. 数据展示组件

在学习 JavaScript 的过程中，我们常常会使用 Swiper 组件来实现轮播图效果。现在，借助 Element Plus 提供的走马灯组件，我们可以更高效地达成相同的目标。

打开 Element Plus 中文官网，在组件页面左侧列表框中找到“Data 数据展示”组件中的走马灯组件。选择其中一个示例代码，将其复制到 index.html 文件的<el-main>标签中，并进行修改，代码如下。

```
<el-main>
  <el-carousel type="card">
    <el-carousel-item>
      <img src="./static/images/carousel01.png" />
    </el-carousel-item>
    <el-carousel-item>
      <img src="./static/images/carousel02.png" />
    </el-carousel-item>
    <el-carousel-item>
      <img src="./static/images/carousel03.png" />
    </el-carousel-item>
  </el-carousel>
</el-main>
```

需要注意的是，直接导入的走马灯组件默认不包含图片，因此不会立即显示轮播效果。为了实现轮播功能，我们可以先将轮播图片放到 static/images/目录下，然后在代码中正确配置图片的引用路径。完成这些步骤后，走马灯的轮播效果即可成功呈现，如图 1-21 所示。

图 1-21　走马灯的轮播效果

4. 表单组件

表单是用户与应用程序交互的重要方式之一。Element Plus 作为一款功能强大的 UI 框架，提供了丰富且易于使用的表单组件，能够帮助开发者快速构建美观、高效且功能丰富的表单界面。Element Plus 的表单组件基于 Vue 3 的响应式系统和组件化开发模式，提供了完整的表单解决方案。它不仅涵盖了常见的表单元素，如输入框、下拉选择框、单选框、多选框、日期选择器等，还提供了强大的表单验证功能，可以轻松实现对用户输入数据的验证和校验。

在每个 form 组件中，都需要一个 form-item 字段作为输入项的容器，用于获取值与验证值。以最简单的登录表单为例，代码如下。

```
<el-form :model="formData">
  <!-- 文本输入框 -->
  <el-form-item label="用户名">
    <el-input v-model="formData.username"></el-input>
```

```
      </el-form-item>
      <!-- 密码输入框 -->
      <el-form-item label="密码">
        <el-input type="password" v-model="formData.password"></el-input>
      </el-form-item>
      <!-- 登录按钮 -->
      <el-form-item>
        <el-button type="primary" @click="login">登录</el-button>
      </el-form-item>
    </el-form>
```

5. 反馈组件

Element Plus 框架提供了一套完善的反馈组件体系，帮助开发者在用户操作后即时传递清晰的状态信息，提升交互体验与系统可感知性。反馈组件包括提示、对话框、抽屉、加载、消息提示、通知等具有丰富交互模式的组件。其中，消息提示（在代码中写为 ElMessage）是高频使用的核心组件之一。

消息提示是轻量级全局提示，常用于表单提交、操作确认等场景，支持成功、警告、错误等状态类型，可以自动消失或手动关闭。消息提示组件可以通过以下代码进行使用，当单击“显示提示消息”按钮时，即可弹出提示消息“操作成功!”。

```
<body>
  <div id="app">
    <el-button @click="open">显示提示消息</el-button>
  </div>
  <script>
    // 导入 ElMessage 组件
    const { ElMessage } = ElementPlus;
    const App = {
      methods: {
        open() {
          // 使用 ElMessage 组件
          ElMessage({
            message: "操作成功! ", // 消息文字
            type: "success", // 消息类型
          });
        },
      },
    };
    Vue.createApp(App).use(ElementPlus).mount("#app");
  </script>
</body>
```

通过灵活组合这些组件，开发者能以极简代码实现符合用户认知的交互反馈，确保信息传达高效且不会打断主流程。

在 Element Plus 中文官网中，有许多现成的样式和功能组件可供使用。前面演示的几个示例只是冰山一角。在实际开发中，各种组件都会被频繁使用。在接下来的章节中，我们也

将不断应用该框架中的各种组件。为了更好地掌握 Element Plus，建议大家养成经常查阅官方文档的习惯，充分利用文档中的示例和说明，提升开发效率并拓展组件的使用技巧。

1.4.3 Element Plus 布局

在现代 Web 开发中，前端 UI 框架通常会提供页面布局解决方案。合理的布局不仅能够使页面结构清晰、美观，还能适配不同设备和屏幕尺寸。Element Plus 提供了强大的栅格系统，网格布局和响应式布局是其核心特性。

栅格系统是一种常用的网页布局技术，通过将页面水平分割成等宽的列，页面元素能够方便地排列在列中。Element Plus 的栅格系统基于 Flexbox 实现，能够快速生成响应式布局，轻松适应各种屏幕尺寸，帮助开发者快速高效地构建复杂的页面布局。

1. 网格布局

网格布局是一种常用的网页布局技术，通过将页面水平分割成等宽的列来排列页面元素。Element Plus 框架主要采用栅格系统来实现页面的网格布局。栅格系统采用 24 列布局方式，即每行最多可包含 24 个栅格单元，其布局如图 1-22 所示。

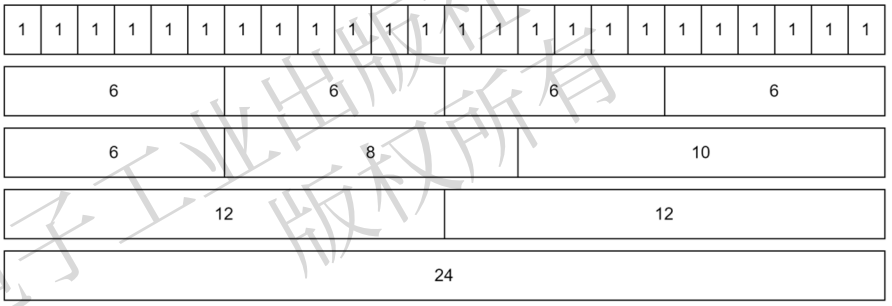

图 1-22　栅格系统的布局

在 Element Plus 的栅格系统中，布局的基本结构由行和列组成。

- row 组件：栅格的行，用于定义一组栅格。
- col 组件：栅格的列，用于定义行中的具体列数。

每行都由 row 组件包裹，而每列则由 col 组件组成。在 row 组件中可以包含多个 col 组件，每个 col 组件可以设置不同的宽度，以实现灵活的自适应布局。可以通过设置 col 组件的 span 属性，指定每个栅格所占的栅格单元数量。例如，:span="12"表示该列占 12 个栅格单元，正好是半行的宽度。由于每一行默认包含 24 个栅格单元，所以所有列的 span 属性值总和不应超过 24。

接下来，我们将学习 Element Plus 栅格系统的一些常见用法，来更好地理解如何使用它来构建页面的网格布局。

1）基础布局

Element Plus 通过 row 组件和 col 组件来定义页面的网格布局。如果要将页面分成两列，第 1 列占 16 个栅格单元，第 2 列占 8 个栅格单元，那么可以使用以下代码来定义。

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Element Plus 网格布局</title>
    <script src="./static/js/vue.global.js"></script>
    <script src="./static/element-plus/index.full.js"></script>
    <link rel="stylesheet" href="./static/element-plus/index.css" />
<style scoped>
      .el-row {
        border: #ccc 1px solid;
      }
      .grid-content {
        min-height: 36px;
        line-height: 36px;
        text-align: center;
      }
      .bg-purple {
        background: #d3dce6;
      }
      .bg-purple-light {
        background: #e9eef3;
      }
    </style>
  </head>
  <body>
    <div id="app">
      <h3>(1)基础布局</h3>
      <el-row>
        <el-col :span="16">
          <div class="grid-content bg-purple">左侧区域 (16)</div>
        </el-col>
        <el-col :span="8">
          <div class="grid-content bg-purple-light">右侧区域 (8)</div>
        </el-col>
      </el-row>
    </div>
    <script>
      const App = {
        data() {
          return {};
        },
      };
      Vue.createApp(App).use(ElementPlus).mount("#app");
    </script>
  </body>
</html>
```

在这个例子中，页面被划分成两列，分别占 16 个和 8 个栅格单元，页面效果如图 1-23 所示。

图 1-23　栅格系统基本布局的页面效果

2）分栏间隔

在默认情况下，栅格是紧贴在一起的。Element Plus 通过 row 组件的 gutter 属性，设置列与列之间的间隔。例如，通过设置:gutter="20"为每个列增加 20px 的间隔，代码如下。

```
<h3>（2）分栏间隔</h3>
<el-row :gutter="20">
  <el-col :span="12">
    <div class="grid-content bg-purple">第 1 列</div>
  </el-col>
  <el-col :span="12">
    <div class="grid-content bg-purple-light">第 2 列</div>
  </el-col>
</el-row>
```

每个列之间会有 20px 的间隔，gutter 属性的值会在左右两侧均匀分布，页面效果如图 1-24 所示。

图 1-24　栅格系统分栏间隔的页面效果

3）列偏移

如果某些列的内容不靠左对齐，而是向右偏移，那么可以通过 col 组件的 offset 属性实现。offset 属性的值指定列在左侧需要偏移的列数。例如，假设第 1 列占用 6 格并偏移 6 格，第 2 列占用 12 格，那么可以使用以下代码实现。

```
<h3>（3）列偏移</h3>
<el-row>
  <el-col :span="6" :offset="6">
    <div class="grid-content bg-purple">偏移 第 1 列</div>
  </el-col>
  <el-col :span="12">
    <div class="grid-content bg-purple-light">第 2 列</div>
  </el-col>
</el-row>
```

此时，第 1 列将向右偏移 6 格，而第 2 列正常显示，页面效果如图 1-25 所示。

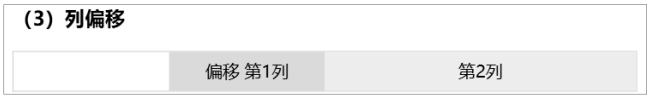

图 1-25　栅格系统列偏移的页面效果

4）Flex 布局支持

row 组件默认使用 Flex 布局对分栏进行灵活对齐，可以通过设置 row 组件的以下属性来控制 Flex 布局的表现。

- type="flex"：启用 Flex 布局。
- justify：设置主轴方向上的对齐方式，支持 start（默认）、center、end、space-around、space-between 和 space-evenly。
- align：设置交叉轴上的对齐方式，支持 top（默认）、middle 和 bottom。

例如，下面的代码将两个列元素居中对齐，并在主轴上均匀分布。

```
<h3>（4）Flex 布局支持</h3>
<el-row type="flex" justify="space-between" align="middle">
  <el-col :span="8">
    <div class="grid-content bg-purple">第 1 列</div>
  </el-col>
  <el-col :span="8">
    <div class="grid-content bg-purple-light">第 2 列</div>
  </el-col>
</el-row>
```

页面效果如图 1-26 所示。

图 1-26 栅格系统 Flex 布局支持的页面效果

通过以上例子，我们可以看到 Element Plus 网格布局是基于栅格系统的 24 列布局，它借助 row 组件和 col 组件实现对基础布局、分栏间隔、列偏移及 Flex 布局的支持，为开发者提供了灵活的页面对齐方式，帮助开发者快速构建复杂的页面布局。

2. 响应式布局

Element Plus 的栅格系统参照了 Bootstrap 的响应式设计，支持页面的响应式布局，可根据浏览器视口（viewport）宽度自动调整列的显示方式。通过为列设置 xs、sm、md、lg、xl 等属性，可分别定义不同视口宽度范围内的列宽和间距，实现灵活的响应式设计。响应式布局属性和对应的视口宽度范围如表 1-1 所示。

表 1-1 响应式布局属性和对应的视口宽度范围

属性	视口宽度范围
xs	宽度<768px 的超小视口
sm	768px≤宽度<992px 的小视口
md	992px≤宽度<1200px 的中视口
lg	1200px≤宽度<1920px 的大视口
xl	宽度≥1920px 的超大视口

我们可以通过为不同宽度范围的视口设置不同的列宽，实现响应式布局。例如，在超小视口中，列占 24 个栅格单元；在小视口或中视口中，列各占 12 个栅格单元；在大视口和超大视口中，列各占 6 个栅格单元，实现代码如下。

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Element Plus 响应式布局</title>
    <script src="./static/js/vue.global.js"></script>
    <script src="./static/element-plus/index.full.js"></script>
    <link rel="stylesheet" href="./static/element-plus/index.css" />
    <link rel="stylesheet" href="./static/element-plus/display.css" />
    <style scoped>
      .grid-content {
        min-height: 36px;
        line-height: 36px;
        text-align: center;
      }
      .bg-purple {
        background: #d3dce6;
      }
      .bg-purple-light {
        background: #e9eef3;
      }
    </style>
  </head>
  <body>
    <div id="app">
      <h3>响应式布局</h3>
      <el-row>
        <el-col :xs="24" :sm="12" :lg="6">
          <div class="grid-content bg-purple">第 1 列</div>
        </el-col>
        <el-col :xs="24" :sm="12" :lg="6">
          <div class="grid-content bg-purple-light">第 2 列</div>
        </el-col>
        <el-col :xs="24" :sm="12" :lg="6">
          <div class="grid-content bg-purple">第 3 列</div>
        </el-col>
        <el-col :xs="24" :sm="12" :lg="6">
          <div class="grid-content bg-purple-light">第 4 列</div>
        </el-col>
      </el-row>
    </div>
```

```
    <script>
      const App = {
        data() {
          return {};
        },
      };
      Vue.createApp(App).use(ElementPlus).mount("#app");
    </script>
  </body>
</html>
```

在浏览器中打开页面，按下键盘上的 F12 键打开浏览器控制台，单击“显示/隐藏设备工具栏”图标打开尺寸调节输入框，输入相应视口宽度即可查看效果，页面效果如图 1-27、图 1-28 和图 1-29 所示。

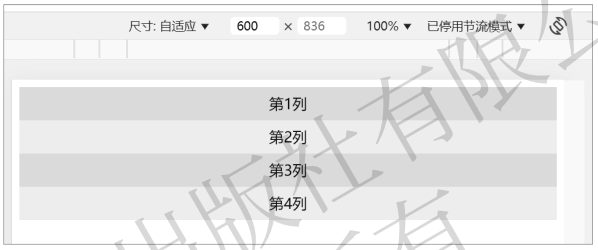

图 1-27　在超小视口中的页面效果（1）

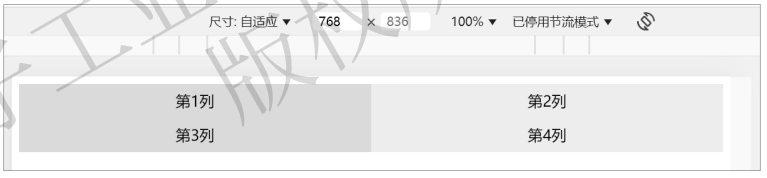

图 1-28　在小视口和中视口中的页面效果（1）

图 1-29　在大视口和超大视口中的页面效果（1）

除此之外，Element Plus 还提供了一系列类名，用于在某些条件下隐藏元素，即基于断点的隐藏类。这些类名可以添加在任何 DOM 元素或自定义组件上。如果读者需要，那么可以自行引入 element-plus/theme-chalk/display.css 文件。

Element Plus 通过为每个宽度范围的视口设置不同的显示/隐藏规则，实现响应式内容控制。例如，在超小视口中仅显示第 1 列，隐藏第 2 列；在小视口和中视口中仅显示第 2 列，隐藏第 1 列；在大视口和超大视口中同时显示第 1 列和第 2 列，实现代码如下。

```
    <el-row>
      <el-col :lg="12" class="hidden-sm-only hidden-md-only">
        <div class="grid-content bg-purple">第 1 列</div>
```

```
      </el-col>
      <el-col :lg="12" class="hidden-xs-only">
        <div class="grid-content bg-purple-light">第 2 列</div>
      </el-col>
    </el-row>
```

页面效果如图 1-30、图 1-31 和图 1-32 所示。

图 1-30　在超小视口中的页面效果（2）

图 1-31　在小视口和中视口中的页面效果（2）

图 1-32　在大视口和超大视口中的页面效果（2）

习题 1.4

一、单选题

1. Element Plus 是专为（　　）版本的 Vue 进行重构的 UI 框架。

A. Vue 2.x　　B. Vue 3.x　　C. React 17　　D. Angular 9

2. Element Plus 的安装方式不包括（　　）。

A. 浏览器直接引入　　B. 使用包管理器（如 NPM、Yarn 或 pnpm）

C. 通过 FTP 上传到服务器　　D. 通过 CDN 链接引入

3. 在 Element Plus 中，以下（　　）组件不属于基础组件类别。

A. 按钮　　B. 图标　　C. 布局容器　　D. 走马灯

4. Element Plus 的栅格系统是基于（　　）布局技术实现的。

A. CSS Grid　　B. Flexbox　　C. CSS Table　　D. CSS Float

5. 在 Element Plus 中，如果想让一个列占据半行的宽度，那么应该将该列的 span 属性值设置为（　　）。

A. 12　　B. 24　　C. 16　　D. 8

6. 在 Element Plus 中，设置列与列之间的间隔的属性是（　　）。

A. gutter　　B. span　　C. offset　　D. align

7. 在 Element Plus 中，设置列偏移的属性是（　　）。

A. offset　　B. gutter　　C. span　　D. justify

8. 在 Element Plus 中，设置（　　）可以启用行（row）组件的 Flex 布局。

A. type="flex"　　B. type="grid"

C. flex="true"　　D. type="row"

9. 在 Element Plus 的响应式布局中，以下（　　）属性用于设置在浏览器中视口（992px≤宽度<1200px）上的列宽。

A. xs　　B. sm　　C. md　　D. lg

10. 在 Element Plus 中，以下（　　）类名用于在浏览器视口宽度小于 768px 时隐藏元素。

A. hidden-xs-only　　B. hidden-sm-and-down

C. hidden-md-and-up　　D. hidden-lg-only

二、判断题

1. Element Plus 允许通过设置 el-col 组件的 span 属性来定义列占据的栅格数，但这个属性的值可以超过 24。（　　）

2. 在 Element Plus 中，el-row 组件的 justify 属性可以控制列在主轴方向上的对齐方式。（　　）

3. Element Plus 的栅格系统通过 row 组件的 offset 属性来设置列与列之间的间隔。（　　）

4. 在 Element Plus 中，可以通过设置列的 xs、sm、md、lg、xl 属性来实现响应式布局。（　　）

5. 在 Element Plus 中，可以使用 hidden-xs-only 类名来在浏览器小视口中隐藏元素。（　　）

任务 1.4　开发网站首页

通过开发“HTML 版技术社区网站”的首页，掌握 Element Plus 核心组件应用与页面布局能力，完成具备动态交互功能的完整页面架构。具体任务要求如下。

1. 环境配置与框架集成

在 HTML 版技术社区网站项目的 static/目录下新建 element-plus 目录，存放从官方 CDN 下载的 Element Plus 样式文件 index.css 和完整脚本文件 index.full.js。

在 index.html 文件中导入 Element Plus 样式文件和脚本文件。

初始化 Vue 应用实例并集成 Element Plus。

2. 首页布局开发

头部区域使用顶栏容器；左侧显示网站 Logo 和名称；中部使用菜单组件实现导航栏，包

含首页、文章、课程、商城四个菜单项；右侧使用按钮组件实现登录和注册按钮。内容区域使用走马灯组件，实现轮播效果。底部区域使用底栏容器，展示版权信息和联系方式等。

3. 设计还原与验收标准

完整实现设计稿的页面布局，以及间距、图标等视觉细节，首页的页面效果如图 1-33 所示。

图 1-33　首页的页面效果

轮播图支持自动播放与手动切换，无图片拉伸变形问题。

在用户单击导航栏菜单项后，该菜单项高亮显示，“登录”和“注册”按钮悬停状态符合交互规范。

通过本任务实践，读者将掌握门户架构基本设计能力，实现工程化组件开发范式，理解项目资源管理规范。

1.5　版本控制与代码托管

在软件开发中，版本控制和代码托管是不可或缺的技能。版本控制能够帮助开发者管理代码的变更历史，而代码托管则提供了代码存储和协作平台。本节将介绍版本控制工具 Git、代码托管平台 Gitee 的使用方法。

1.5.1 版本控制工具 Git

版本控制是软件开发的重要环节之一，它允许开发者跟踪代码的变更历史、回滚到之前的版本、管理不同的开发分支，以及协作开发。Git 是目前流行的分布式版本控制系统，它提供了强大的功能，可用于管理代码的版本和变更。

1. Git 安装

访问 Git 官网，根据计算机系统选择对应的安装包，下载后按照默认设置安装即可。

在安装完成后，右击计算机任意位置，在弹出的快捷菜单中选择“Git Bash here”命令，即可打开 Git 命令行工具 Git Bash。

2. Git 基本概念

仓库（Repository）：存储项目的所有文件和变更历史。

分支（Branch）：独立的开发线，允许开发者在不影响主代码库的情况下进行实验和开发。

提交（Commit）：将代码的变更保存到仓库中，每个提交都有一个唯一的标识符。

合并（Merge）：将一个分支的变更合并到另一个分支中。

拉取（Pull）：从远程仓库获取最新的变更并合并到本地仓库中。

推送（Push）：将本地仓库的变更推送到远程仓库中。

3. Git 基本操作命令

Git 基本操作命令如表 1-2 所示。

表 1-2 Git 基本操作命令

命令	作用
git init	初始化仓库
git add <file>	添加文件到暂存区
git commit -m "提交信息"	提交变更
git status	查看状态
git log	查看提交历史
git branch <branch-name>	创建分支
git checkout <branch-name>	切换分支
git merge <branch-name>	合并分支
git pull	拉取远程仓库最新代码
git push	推送到远程仓库

1.5.2 代码托管平台 Gitee

Gitee 是一个基于 Git 的代码托管和团队协作平台，提供了代码存储、项目管理、代码审查、持续集成等功能。Gitee 支持多种编程语言和框架，非常适合团队协作开发。我们借助 VS

Code 编辑器集成 Git 功能，使用 Gitee 托管前端项目代码，并使用 Git 进行版本控制。

基本操作步骤如下。

1. 注册 Gitee 账号

访问 Gitee 官网，单击“注册”按钮，填写相关信息完成注册。

在任意位置打开 Git Bash，输入以下命令，配置全局用户信息，包括用户名和邮箱，以便在 Git 版本控制中记录用户信息。

```
git config --global user.name 'yourname'
git config --global user.email 'youremail'
```

2. 创建仓库

登录 Gitee 账号，在网站右上角找到“+”图标，将鼠标指针移至该图标上方，在弹出的下拉菜单中选择“新建仓库”选项。在打开的“新建仓库”页面中填写仓库名称、路径、仓库介绍等信息，选择公开或私有仓库，还可以勾选“初始化仓库”复选框（勾选后在“.gitignore”下拉列表中选择“Node”模板），完成创建。

3. 克隆仓库

在 Gitee 仓库页面中单击“克隆/下载”按钮，复制仓库的克隆地址。

在 Git Bash 中执行以下命令克隆仓库。

```
git clone <仓库地址>
```

在克隆到本地后，将之前创建的项目文件复制到克隆的仓库目录下。注意，不要覆盖.git 隐藏文件夹。

4. 推送到 Gitee

首先，在本地仓库中执行以下暂存命令。

```
git add .
```

该命令可以将当前目录及其子目录下的所有更改（包括新文件、修改的文件和删除的文件）添加到暂存区。

然后，执行以下提交命令。

```
git commit -m "提交信息"
```

该命令会将暂存区的更改提交到本地仓库中，并添加提交信息。提交信息应该简洁明了，描述清楚本次提交的主要内容和目的，如果提交信息过于模糊或不具体，那么可能会给后续的代码审查和维护带来困难。

最后，执行以下推送命令。

```
git push
```

该命令会将本地仓库的更改推送到远程仓库中。如果远程仓库有新的更改，而本地仓库没有拉取这些更改，那么可能会导致推送失败或产生冲突。如果远程仓库的分支保护规则不允许直接推送，那么也可能会导致推送失败。

除了使用以上命令，我们还可以通过 VS Code 内置的 Git 工具完成代码推送。方法如下：打开 VS Code 源代码管理窗口，在消息输入框中输入提交信息（一定要输入提交信息，否则无法提交），单击“提交”按钮，此时 Git 会暂存所有更改，并提交到本地仓库中。当首次提交时，会弹出警告对话框，提示“没有可提交的暂存更改。是否要暂存所有更改并直接提交？”单击“总是”按钮即可，下次提交就不会出现该警告了。

5. 拉取 Gitee 仓库的变更

在本地仓库中执行拉取命令，拉取 Gitee 远程仓库的最新变更。

```
git pull
```

我们也可以在 VS Code 源代码管理窗口中单击“同步”图标，如图 1-34 所示，即可同时进行推送和拉取。

图 1-34 单击“同步”图标

6. 创建和管理分支

在 Gitee 仓库页面中选择“分支”标签，进入分支管理页面，可以在此页面中查看和管理分支。

在本地 Git Bash 中执行以下命令创建和切换分支。

```
git branch <branch-name>
git checkout <branch-name>
```

也可以执行以下命令，创建分支并切换到该分支。

```
git checkout -b <branch-name>
```

执行以下命令可以推送本地分支到远程仓库中。

```
git push origin <branch-name>
```

7. 合并分支

在 Gitee 仓库分支管理页面中单击“创建 Pull Request”按钮，创建一个新的合并分支请求。

在本地 Git Bash 中执行以下命令合并分支。

```
git checkout master
git merge <branch-name>
git push
```

本节介绍了如何通过 Git 和 Gitee 对本地前端项目进行版本控制和代码托管，确保团队协作开发的高效性和代码的稳定性。

习题 1.5

一、单选题

1. Git 是一种（　　）。

A. 集中式版本控制系统　　B. 分布式版本控制系统

C. 网络式版本控制系统　　D. 单机式版本控制系统

2. 在 Git 中，用于将代码变更保存到仓库中的命令是（　　）。

A. git add　　B. git commit　　C. git push　　D. git pull

3. 在 Gitee 上创建仓库时，以下（　　）步骤是正确的。

A. 单击“注册”按钮，填写相关信息完成注册

B. 选择“新建仓库”选项，填写仓库名称、路径等信息

C. 单击“克隆/下载”按钮，复制仓库的克隆地址

D. 单击“合并请求”按钮，创建一个新的合并请求

4. 在 Git Bash 中，用于克隆仓库的命令是（　　）。

A. git clone <仓库地址>　　B. git pull <仓库地址>

C. git push <仓库地址>　　D. git add <仓库地址>

5. 在 Git 中，用于从远程仓库获取最新变更并合并到本地仓库的命令是（　　）。

A. git add　　B. git commit　　C. git pull　　D. git push

6. 在 Git 中，用于将本地仓库的变更推送到远程仓库的命令是（　　）。

A. git add　　B. git commit　　C. git pull　　D. git push

二、判断题

1. Git 是一种集中式版本控制系统。（　　）

2. Gitee 是一个基于 Git 的代码托管平台，提供代码存储、项目管理和代码审查等功能。（　　）

3. 在 Git 中，分支（Branch）指独立的开发线，允许开发者在不影响主代码库的情况下进行实验和开发。（　　）

4. Git 的 git commit 命令用于将代码的变更保存到仓库中，并且每个提交都有一个唯一的标识符。（　　）

任务 1.5　实现代码托管

通过实现“HTML 版技术社区网站”的项目代码托管，掌握 Git 版本控制核心工作流，完成本地项目到 Gitee 远程仓库的全流程托管。具体任务要求如下。

1. 环境初始化

（1）注册 Gitee 账号，完成邮箱验证。

（2）安装 Git 客户端，配置全局用户信息。

2. 远程仓库管理

（1）在 Gitee 中创建仓库，命名为“html-tech”。

（2）克隆仓库到本地。

3. 项目迁移与版本控制

（1）将本地 tech 项目文件复制到克隆的仓库目录中。

（2）提交并推送代码变更到远程仓库中。

4. 协作验证与规范

（1）登录 Gitee 查看仓库推送记录，确认文件完整上传。

（2）模拟团队协作，拉取远程最新代码。

（3）在本地仓库中创建 dev 开发分支并推送到远程仓库中，实践分支管理流程。

通过本任务实践，读者将具备企业级代码管理能力，掌握持续集成的基础配置方法，形成工程化协作思维模式，具备团队协作开发基础能力。